An Introduction to Bioconductor

Bethany Wolf

Statistical Computing I

January 26, 2012

Overview

- Background on Bioconductor project
- Installation and Packages in Bioconductor
- An example: working with microarray meta-data

Bioconductor

- Biological experiments continually generate more data and larger datasets
- Analysis of large datasets is nearly impossible without statistics and bioinformatics
- Research groups often re-write the same software with slightly different purposes
- Bioconductor includes a set of open-source/open-development tools that are employable in a broad number of biomedical research areas

Bioconductor Project

- The Bioconductor project was started in 2001 with goal of making it easier to conduct reproducible consistent analysis of data from new high-throughput biological technologies
- The core maintainers of the Biocoductor website are located at Fred Hutchinson Cancer Research Center
- Updated version released biannually and the release coincides with the release of R
- Like R, there are contributed software packages

Goals of the Bioconductor project

- Provide access to statistical and graphical tools for analysis of high-dimensional biological data
  - micro-array analysis
  - analysis of high-throughput
- Include comprehensive documentation describing and providing examples for packages
  - Website provides sample workflows for different types of analysis
  - Packages have associated vignettes that provide examples of how to use functions
- Have additional tools to work with publically available databases and other meta-data

Bioconductor website

- Lets take a look at the Bioconductor website...

  http://bioconductor.org/

Installing Bioconductor

- All packages available in bioconductor are run using R
- Bioconductor must be installed within the R environment prior to installing and using Bioconductor packages

```
> source("http://bioconductor.org/biocLite.R")
> biocLite()
```

Bioconductor Packages

- Biobase is the base package installed when you install bioconductor
- It includes several key packages (e.g. affy and limma) as well as several sample datasets

```
> biocLite("Biobase")
> library(Biobase)
```

Help Files for Bioconductor Packages

- Like R, there are help files available for Bioconductor packages. They can be accessed in several ways.

```
> help(Biobase)
> library(help="Biobase")
> browseVingettes(package="Biobase")
> Use the Vignettes pull down menu in R
```

- Note vignettes often contains more information than a traditional R help page.

Other Bioconductor Package

- Packages are similar to R packages and are loaded into and used in R
- However, Bioconductor makes more use of the S4 class system from R where as R packages typically use the S3 class system. The difference...
    - S4 more formal and rigorous (makes it somewhat more complicated than R)
- If you really want to know more about the S4 class system you can check out http://cran.r-project.org/doc/contrib/Genolini-S4tutorialV0-5en.pdf

Some Interesting Bioconductor Packages

Analysis/Data Types	**Package Name**
General Instructions	Biobase, Biostrings, biocViews
Pre-processing Affy/oligo data	affy, affycomp, oligo, makecdfenv
Pre-processing 2-channel microarrays	vsn limma
Differential Gene Expression	edd, genefilter, limma, ROC, siggenes
GSEA/Hypergeometric Testing	GSEABase, Category, GOstats, topGO
Annontation	annotate, annaffy, AnnotationDbi
Graphs and Networks	graph, RBGL, Rgraphviz
High-throughput Sequencing Data	BSgenome, ShortRead, HilbertVis
Flow Cytometry	prada, flowCore, flowViz, flowUtils
Protein Interactions	ppiData, ppiStats, ScISI, Rintact
Other data	xcms, DNAcopy, PROcess, aCGH

Microarray Experiment

- Microarrays are collections of microscopic DNA spots attached to solid surface
- Spots contain probes, i.e. short segments of DNA gene sections
- Probes hybridize with cDNA or cRNA in sample (targets)
- Fluorescent probes used to quantify relative abundance of targets
- Can be used to measure expression level, change in expression, SNPs,...

Gene Detection

- 1-Channel array: hybridized cDNA from a single sample to the array and measure intensity to determine gene expression.
  - label sample with a single fluorophore
  - compare relative intensity to a reference sample done on a separate chip
- 2-Channel arrays: hybridized cDNA for two samples (e.g. diseased vs. healthy tissue)
  - label each with one of two different fluorophores
  - mix two samples and apply to single microarray
  - look at fluoresence at 2 wavelengths corresponding to each fluorophone
  - measure ratio of intesity for each fluorophore

Microarray Analysis

- Microarrays are large datasets that often have poor precision
- Statistical challenges
  - Account for effect of background noise
  - Data normalization (remove non-biological variability)
  - Detecting/removing poor quality or low quality feature (flagging)
  - Multiple comparisons and clustering analysis (e.g FDR, hierarchical clustering)
  - Network analysis (e.g. Gene Ontology)

Example Meta-data in Bioconductor

- sample.ExpressionSet is an example of microrray meta-data provided in Biobase
- It is of class ExpressionSet (example of an S4 class). This class includes data describing the lab, the experiment, and an abstract that are all accessible in R.

```
> data(sample.ExpressionSet)
> sample.ExpressionSet
```

Exploring sample.ExpressionSet

- What information can we find out about the meta-data sample.ExpressionSet

```
> class(sample.ExpressionSet)
> slotNames(sample.ExpressionSet) #Attributes of the meta-data
```

Difference from S3 class object

- So how different is this from a S3 class object? Linear models fit using lm are S3 class objects for example.

```
> x<-rnorm(100); y<-rnorm(100); fit<-lm(y x)
> class(fit)
> slotNames(fit)
> names(fit); fit$coefficients
> class(sample.ExpressionSet)
> slotNames(sample.ExpressionSet)
> sample.ExpressionSet@experimentData
```

- So we can access attributes in an S3 object using "$" and in an S4 object using "@".

Accessing and Expression Set

- Accessing data and parts of the data using the "@" symbol can be dangerous
- **R** does not provide a mechanism for protecting data (i.e. we can overwrite our data by accident)
- A better idea is to subset the parts of the data you want to handle

Exploring sample.ExpressionSet

- Although slotNames tells us what attributes sample.ExpressionSet has, we are interested in accessing the microarray data itself.

```
> abstract(sample.ExpressionSet)
> varMetadata(sample.ExpressionSet) #Variable names of the data
> featureNames(sample.ExpressionSet) #Names of the genes
> exprs(sample.ExpressionSet) #Expression values for the genes
```

Visualizing the Data

- Let's look at the distribution of gene expression values for all of the arrays.

```
> dim(sample.ExpressionSet)
> plot(density(exprs(sample.ExpressionSet)[,1]), xlim=c(0,6000),
  ylim=c(0, 0.006), main="Sample densities")
> for (i in 2:25)
> lines(density(exprs(sample.ExpressionSet)[,i]), col=i)
```

Subsetting the data

We can subset our microarray object just like a matrix. Note the in gene array datasets, samples are columns and features are rows. Thus if we want to subset of samples (i.e. things like cases or controls) we want columns. However if we are interested in particular probes, we subset on rows.

```
> sample.ExpressionSet$sex #Gender all 26 microarrays
> subESet<-sample.ExpressionSet[1:10,] #All information on the first 10 genes
> exprs(sample.ExpressionSet)[1:10,] #Expression for first 10 genes
```

Subsetting the data

- What if we only want to consider females?

```
> f.ids<-which(sample.ExpressionSet$sex=="Female")
> femalesESet<-sample.ExpressionSet[,f.ids]
```

- What if we only want to only AFFX genes? We can use the command grep in this case...

```
> AFFX.ids<-grep("AFFX", featureNames(sample.ExpressionSet))
> AFFX.ESet<-sample.ExpressionSet[,AFFX.ids]
```

Next Steps?

- Now we are familiar with the data, we could go the next step and do the analysis...
  1. Pre-processing: assess quality of the data, remove any probes we know to be non-informative
  2. Look for differential expression using a machine learning technique
  3. Annotation
  4. Gene set enrichment
  5. ...

- Fortunately, bioconductor provides workflows for many common analyses to help you get started.

  http://bioconductor.org/workflows

- Although R has many statistical packages, packages in Bioconductor are designed for bioinformatics type problems
- We have only touched on one small part of what is available
- For further help using bioconductor
  - The Bioconductor website has workshops from previous years
  - There is also an annual User's group meeting
  - Package vignettes and help files also often contain examples with "real" data so you can work through and example